form a moved member. The moved member is combined with a second member associated with the pointer. This process is repeated until all the fourth data string is processed.

FIG. 10 is a flow chart of the steps used in an untransform function in accordance with one embodiment of the invention. The untransform module can determine the first transform for a first data string given the first-second transform and the second data string. The process starts, step 220, by extracting the most significant portion 222 of the first-second transform 224 at step 226. The most significant portion 222 is a reverse pointer that is associated with a pointer 228 in the reverse look-up table. The pointer is accessed at step 230. Next the first-second transform 224 is combined with a member 232 associated with the pointer to form an intermediate product 234 at step 236. The intermediate product is moved left by the number of bits in the pointer 228 at step 238. This forms a moved intermediate product 240. Next the pointer 228 is combined with the second data string 242 to form a result 244 at step 246. The result 244 is combined with the moved intermediate product 240 to form the first transform 248 at step 250, which ends the process at step 252. Again this module is repeated multiple times if the second data string is longer than the pointer.

Some examples of what the transform module 100 can do, include determining a second-third transform from a first-second-third transform and a first transform. The first transform is shifted by the number of data bits in the second-third data string. The shifted first transform is combined with the first-second-third transform to form the second-third transform. In another example, the transform generator

could determine a first-second-third-fourth transform after receiving a fourth data string. In one example, the transform module would first calculate the fourth transform (using the transform module). Using the shift module the first-second-third transform would be shifted by the number of data bits in the forth data string. Then the shifted first-second-third transform is combined, using the combiner, with the fourth transform.

FIG. 13 is a block diagram of a system 270 for associative processing in accordance with one embodiment. The system 270 has an icon generator 272. The icon generator 272 has an input 274 connected to key data or input data that is converted to icons. The icon generator is connected to an associative memory controller 276. The associative memory controller (AMC) 276 receives icons from the icon generator 272. The associative memory controller 276 is connected to a RAM (random access memory; memory) 278. The AMC 276 and the RAM 278 form a virtual associative memory. The AMC 276 is connected to an associative processing unit 280. Note that the icon contains an address and a confirmer. The address is used to access the RAM 278 by the AMC 276. A confirmer from the address in the RAM is compared to the confirmer of the icon determine if a match has been found. For more information on the use of addresses and confirmers see USPN 5,942,002 and US patent application serial No. 09/419,217 both assigned to the same assignee as the present application and hereby incorporated by reference.

The icon generator may use a polynomial code to convert the key into an icon (or hash). The icon generator may also produce a plurality

of lengths of icons. For more details on how the icon generator can produce multiple lengths of icons see US patent application entitled "Method of Forming a Hashing Code", serial no. 09/672,754, filed on September 28 2000 assigned to the same assignee as the present application and hereby incorporated by reference. The hardware to produce the icon may be linear feedback shift register (See FIG. 14) as used to produce CRCs (cyclical redundancy code). Or may be a microprocessor running the algorithms shown in FIGs. 9 & 12. Note that FIG. 12 is a lookup table.

The associative memory controller 276 may be a microprocessor that controls the functions of the RAM, such as lookups, stores, deletes, and comparing of confirmers. This list is not meant to be exhaustive just exemplary. The associative processing unit 280 may be a microprocessor. In addition the APU 280 may include shift registers and exclusive OR arrays. Among the functions the APU 280 might perform are the shift module, unshift module and untransform module shown in FIGs 7, 8 & 10. In addition, any icon algebra that may be necessary. A formal treatment of the icon or linear algebra the APU 280 may perform is given in the appendix of the provisional patent application having serial no. 09/767,493, entitled "Definition of Digital Pattern Processing" filed on October 13, 2000, and assigned to the same assigned as the present application and providing priority for the present application. A less formal and less complete treatment of the icon algebra is discussed in USPN 5,942,002. In one embodiment, a single microprocessor may perform the functions of the IG 272, AMC 276 and APU 280.